Computer Science 350
Assignment Three

Steven Kerr 6022796

21/05/2019

1. Consider the language: $L = \{(M, w) : M$ is a TM and M accepts w or M is undefined on $w\}$.

a) For every Turing machine there are 3 possibilities:

(a) The turing machine accepts the language and halts.

(b) The turing machine rejects the language and halts.

(c) The turing machine is undefined on the language and loops forever.

As we can see, The language L consists of Turing machines that accept w and turing machines that are undefined on w. This means that for all $M \in L$ M rejects x iff M stops on x and rejects x.

b) $\overline{L} = \{< M, w > \mid M$ is a TM and M rejects w $\}$

c) Let us define turing recognisable. This means, there exists a turing machine M such that, on a given language, M will halt and accept only the strings in the language. For any strings not in the language M will either reject or not halt at all. Let us design a turing machine $\overline{M}$ that recognises $\overline{L}$ in the following way:
$\overline{M}$ = 'On input $< M, w >$', an encoding of a TM M and a string w:

1) Use the description of M to construct TM M_1.

2) Run M_1 on w.

3) If M_1 rejects w, *accept*

As we can see above, $\overline{M}$ recognises only the TM's M such that M has a reject state. Since we are only concerned with recognisability, this is the only requrement that needs to be met. All other strings are either rejected or loop for ever.

2. Let $L = \{< M >: M \text{ is a } TM \text{ that accepts atleast two binary strings}\}$.

a) A language is recognisable if there exists a turing machine such that, on a given language, will halt and accept only the strings in the language. For any strings not in the language will either reject or not halt at all. For an undecidable language, there is no Turing Machine which accepts the language and makes a decision for every input string w. The halting problem is an example of undecidability.

b) Informally speaking, I think L is turing-recognisable because is could design a machine M^b the checks to see if there exists two or more paths from the starting state to an accept state of given $< M >$.

c) Let P be a language consisting of TM descriptions where P fulfils two conditions.

1) P is nontrivial, it contains some, but not all, TM descriptions.
2) Whenever $L(M_1) = L(M_2)$ we have $< M_1 > \in P$ iff $< M_2 > \in P$. Here M_1 and M_2 are any TMs.

Then, P is undecidable.
Assume for the sake of contradiction that P is decidable and let R_p be a TM that decides P. So, let $T_\emptyset$ be a turing machine such that $L(T_\emptyset) = \emptyset$ and assume $< T_\emptyset > \notin P$. Because P is not trivial, there exists a TM T with $< T > \in P$.
The following TM S decides $HALT_{TM}$.
S = 'On input $< M >$':

1) Use M to construct the following TM M'.
   M' = 'On input x and y where x and y are binary strings with length ≥ 2 and $x \neq y$:
   1) Simulate M on input x. If it halts and rejects, *reject*. If it accepts, proceed to stage 2.
   2) Simulate M on input y. If it halts and rejects, *reject*. If it accepts, proceed to stage 3.
   3) Simulate T on x. If it accepts go to 4.
   4) Simulate T on y. If it accepts, *accept*
2) Use TM R_p to determine whether $< M' > \in P$. If YES, *accept*. If NO, *reject*.

As we can see above M' simulates T if M accepts x and y, hence $L(M') = L(T)$. Therefore $< M > \in P$ iff M accepts x and y.

Now, let The following TM $S^{'}$ decide $\overline{HALT_{TM}}$.
$S^{'}$ = 'On input $< M >$':

1) Use M to construct the following TM $M^{''}$.
   $M^{''}$ = 'On input x and y where x and y are binary strings with length ≥ 2 and $x \neq y$:
   1) Simulate M on input x. If it accepts, *reject*, If it rejects proceed to stage 2.
   2) Simulate M on input y. If it accepts, *reject*, If it rejects proceed to stage 3.
   3) Simulate $T_{\emptyset}$ on x. If it rejects go to 4.
   4) Simulate $T_{\emptyset}$ on y. If it rejects, *accept*
2) Use TM R_p to determine whether $< M^{''} > \notin P$. If YES, *accept*. If NO, *reject*.

As we can see above $M^{''}$ simulates $T_{\emptyset}$ if M rejects x and y, hence $L(M^{''}) = L(T_{\emptyset})$. Therefore $< M > \notin P$ iff M rejects x and y.

For Rice's Theorem of undecidability I have satisfied the first part by showing that $\exists M \in P$ and I have showen that $\exists M \notin P$. I have also satisfied the second part by showing that $L(M^{'}) = L(T)$ and that $< M^{'} > \in P$ iff $< T > \in P$. Also showing that $L(M^{''}) = L(T_{\emptyset})$ and that $< M^{''} > \notin P$ iff $< T_{\emptyset} > \notin P$. Therefore, L is undecidable.

d) Bonus Question.
Because we know that $HALT_{TM}$ is recognisable I will start by assuming there exists a turing machine M^* that recognises L and attempt to prove $M^* \leq_m HALT_{TM}$.
We start by constructing a computable function f that takes inputs of the form $< M >$ and returns outputs of the form $< M^{'} >$, where

$$< M > \in M^* \iff < M^{'} > \in HALT_{TM}$$

The following TM F computes f.
F = 'On input $< M >$:

1) Construct the following machine $M^{'}$.
   $M^{'}$ = 'On input x and y where x and y are binary strings with length ≥ 2 and $x \neq y$:
   1) Run M on x.
   2) If M accepts, go to 4.
   3) If M rejects, enter a loop.
   4) Run M on y.
   5) If M accepts, *accept*.
   6) If M rejects, enter a loop.
2) Output $< M^{'} >$.

If machine M does not accept the strings x and y the this will be detected and in such case F outputs a string not in $HALT_{TM}$. Therefore, I have shown, using mapping reducibility that the language L is turing decidable because $HALT_{TM}$ is turing decidable.

3.

a) Define K. K is the descriptive complexity or Kolmogorov-Chaitin complexity of x, written as $K(x)$. consider the following two strings:
S_1 = 'abababababababababababababababab'
S_2 = '4c1j5b2p0cv4w1x8rx2y39umgw5q85s7'
The first string can be described as 'ab 16 times', this consists of 11 charecters. the second has no obvious description other than writing down the string itself, which has 32 charecters. The complexity of a string is the length of the shortest possible description of the string in some fixed language. It can be shown that the Kolmogorov complexity of any string cannot be more than a few bytes larger than the length of the string itself. If M is a Turing Machine which, on input w, outputs string x, then the concatenated string $< M > w$ is a description of x. If a description $d(x)$ of a string x is of minimal length it is called a minimal description of x. Thus, the length of $d(x)$ is the Kolmogorov complexity of x written as $K(x) = |d(x)|$.

b) Prove the exsistance of c such that $K(xxx) \leq K(x) + c$:
Proof: Consider the following TM M, which expects an input of the form $< N, w >$, where N is a TM and w is an input for it.
M = 'On input $< N, w >$:

1) Run N on w until it halts and produces an output string s.
2) Output the string sss.

A description of xxx is $< M > d(x)$. as $d(x)$ is a minimal description of x, the length of this description is:

$$| < M > d(x)| = 2| < M > | + |d(x)| = c + K(x)$$

where $c = 2| < M > |$

c) $K(f(x)) \leq K(x) + c$:
Proof: Every computable function f can be represented by a turing machine F which takes input x and outputs y. A general description language is a computable function $f : \Sigma^* \to \Sigma^*$; the minimal description of x with respect to f, written as $d_f(x)$, is the lexicographically shortest string s where $f(s) = x$ and

$$K_f(x) = |d_f(x)|$$

Using Theorem 6.27 for any descriptional language f, a fixed constant c exists that depends only on f such that

$$\forall x[K(x) \leq K_f(x) + c]$$

Let f be a description language and consider the following TM M: $M =$ 'On input w':

1) Output $f(x)$.

Then $< M > d_f(x)$ is a description of x and:

$$| < M > d_f(x)| = K_f(x) + 2| < M > | = K_f(x) + c$$

where $c = 2| < M > |$

As we can see from TM M and proving that $K(f(x)) \leq K(x) + c$

4.

1. In computability theory, Rice's theorem states that all non-trivial, semantic properties of programs are undecidable. A semantic property is one about the program's behavior. For instance, does the program terminate for all inputs, unlike a syntactic property, for instance, does the program contain an if-then-else statement. A property is non-trivial if it is neither true for every computable function, nor false for every computable function. Another way of stating Rice's theorem that is more useful in computability theory follows.
   Let P be a language consisting of TM descriptions where P fulfils two conditions.

   1) P is nontrivial, it contains some, but not all, TM descriptions.
   2) Whenever $L(M_1) = L(M_2)$ we have $< M_1 > \in P$ iff $< M_2 > \in P$. Here M_1 and M_2 are any TMs.

   Then, P is undecidable.

2. To prove that requirement 1 is essential we will begin by assuming that P is trivial. This means that P contains all TMs or P is empty.
   First, we let P contain nall TMs, then we have $P = \emptyset$. We will construct a recogniser R to recognise P in the following way: $R =$ 'On input $< M, w >$'

   1) If M accepts w then *accept*

2) If M rejects w or goes into a loop, this has no effect on R as it only recognises, not decides.

Therefore, we can say that P is Turing-recognizable.

Second, we construct a Recogniser $\overline{R}$ to recognise $\overline{P}$ where $\overline{P} \neq \emptyset$ in the following way:
$\overline{R}$ = 'On input $< M, w >$'

1) If M rejects w or goes into a loop, *accept*
2) If M accepts w then $reject$

Therefore, we can say that $\overline{P}$ is turing recognisable.

As we can see above, P is co-turing recognisable and from theorem 4.22, 'A language is decidable iff it is turing recognisable and co-turing recognisable' Therefore, P must be 'not trivial'

To prove that regirement 2 is essential we know that whenever $L(M_1) = L(M_2)$, we have $< M_1 > \in P$ iff $< M_2 > \in P$. Requirement 2 is false if we have $L(M_1) = L(M_2)$, and $< M_1 > \in P$ iff $< M_2 > \notin P$
We know that if $< M_2 > \notin P$ then $< M_2 > \in \overline{P}$.
We start by assuming we have a turing machine M that decides P, Then let M_1 recognise P and M_2 recognise $overline P$ in the following way
M = 'On input $< M_1, M_2 >$ and a string w'

1) Run M_1 and M_2 on w in parallel
2) If $w \in P$, if M_1 acceptes w, *accept*. Else, M_1 rejects, $reject$.
3) If $w \in \overline{P}$, if M_2 acceptes w, *accept*. Else, M_2 rejects, $reject$.

As we can see above, P is therefore decidable which is a contradiction to Rice's theorem. Therefore, requirement 2 is essential.